PB93-228708

BAND MATRIX SYSTEMS SOLVERS ON ENSEMBLE ARCHITECTURES

THINKING MACHINES CORP.
CAMBRIDGE, MA

1986

U.S. DEPARTMENT OF COMMERCE
National Technical Information Service

TMC-123

PB93-228708

Band Matrix Systems Solvers on Ensemble Architectures

S.L. Johnsson

Thinking Machines Corporation
Technical Report Series

NA86-3

REPORT DOCUMENTATION PAGE

PB93-228708

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	1986	Technical

4. TITLE AND SUBTITLE
Band Matrix systems solvers on ensemble architectures

5. FUNDING NUMBERS
Office of Naval researc
N00014-84-K-0043

6. AUTHOR(S)
S.L. Johnsson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Thinking Machines Corp.
245 First Street
Cambridge, MA 02142-1264

8. PERFORMING ORGANIZATION REPORT NUMBER
TMC-123

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ONR -- Dept. Navy
The Pentagon
Washington, DC 20350

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We present ditect xolvers for band matrix systems for processor ensembles configured a 2-dimensional meshes with end-around connections, binary trees, shuffle-exchange. perfect shuffle and boolean cube networks, and as clusters of processors with intracluster connectionsforming a torus or a boolean cube and intercluster connections forming binary trees, shuffle-exchange, perfect shuffle and boolean cube networks. The ensembles are assumed to be of the MIMD type, and each processor is equiped with substantial local storage. There is no shared storage, abd control is distributed.

14. SUBJECT TERMS
Numerical Algorithms

15. NUMBER OF PAGES
17

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	SAR	SAR	SAR

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Band Matrix Systems Solvers on Ensemble Architectures

S.Lennart Johnsson
Department of Computer Science
and Electrical Engineering
Yale University
New Haven, CT 06520

NA86-3

Abstract

We present direct solvers for band matrix systems for processor ensembles configured as 2-dimensional meshes with end-around connections, binary trees, shuffle-exchange, perfect shuffle and boolean cube networks, and as clusters of processors with intracluster connections forming a torus or a boolean cube and intercluster connections forming binary trees, shuffle-exchange, perfect shuffle and boolean cube networks. The ensembles are assumed to be of the MIMD type, and each processor is equipped with substantial local storage. There is no shared storage, and the control is distributed. The band matrix system solvers are based on (block) Gaussian elimination, (block) cyclic reduction, or a combination thereof to yield algorithms for banded systems of a minimum complexity $O(m+m\log_2(N/m))$ with mN processors for systems of N equations and bandwidth 2m+1. The banded systems solvers treat the band as being dense.

1. Introduction

Multiprocessor systems with a hundred microprocessors or more are commercially available, and systems with thousands or tens of thousands of processors will become available in the immediate future. A substantially larger number of processors is expected to be both economically and technologically feasible in a computer system within a decade. Fine grain parallelism becomes a possibility even for computations on substantial data sets, and data sets with extensive operations. With a decreasing granularity of computations the interprocessor communication increases in importance, since the complexity of many computations are superlinear in the data sets. For instance, the common matrix multiplication algorithms require order $O(N^3)$ operations for order $O(N^2)$ data elements. The Fast Fourier Transform requires order $O(\log_2 N)$ operations per point, and many sorting algorithms require at least the same order of operations per data element. The most well known, and until recently also the most efficent parallel sorting algorithm, bitonic sort [2], requires $O(\log_2^2 N)$ operations per data element.

Several candidate architectures with different characteristics with respect to performance, ease of programming, scalability with respect to the number of nodes in the system, and scalability with respect

to the technology are currently being investigated. In a much simplified categorization the multiprocessor systems can be considered as switch based systems, direct processor-to-processor connected systems, or hierarchical systems having elements of both these architectures, such as the CEDAR project [25]. Examples of highly concurrent switch based systems are the NYU Ultracomputer [33] [10], and the TRAC [36], and examples of processor-to-processor connected systems are the Caltech Cosmic Cube [35] (and its commercial successor, the INTEL iPSC), the Connection Machine [13], the Caltech Tree Machine [3], and the Columbia NON-VON computer [37], [38]. The ChiP machine [39] is a hybrid between the first two kinds of systems. It has switching elements inserted in the processor-to-processor connections and thereby achieves reconfigurability. All the systems mentioned are of the MIMD (Multiple Instruction streams Multiple Data streams) type [7], with the exception of the NON-VON machine which is of the SIMD (Single Instruction stream Multiple Data streams) type.

For our algorithms we assume that the machines are of the MIMD type. We also assume that the processors are directly interconnected to form a network of processors. The MIMD architectures offers greater flexibility than SIMD architectures and avoids the potential bottleneck and source of limited scalability that a global controller represents, at the expense of possible code duplication and additional hardware. We refer to highly concurrent systems in which processors have their own local storage and instruction streams as *ensemble architectures*. Ensembles of processors directly interconnected to form a network makes the time to access data non-uniform. To take full advantage of such an architecture it is important to exploit the locality that exists in many computations. In VLSI technology the access time to various parts of a system is indeed non-uniform [34] [28], and processor-to-processor interconnected ensembles reflect this property at a higher level. In switch based systems with processors on one side of the switch and storage modules on the other the access time is the same from any processor to any storage module. In the case of an Ω-network the number of routing steps for any processor to processor communication is the same as the maximum routing distance in a boolean cube configured ensemble with the same number of processors.

In this paper we assume that processors are directly interconnected to each other in a sparse and regular manner, and that the architecture is of the MIMD type. These architectures have the potential for exploiting the locality of computations. We focus on the mapping of the computations for the solution of banded systems of equations on to a few prototype ensemble architectures such that the locality of the computations is preserved in order to achieve a low communication (and computational) complexity. Several parallel algorithms for the solution of dense banded systems of equations have been proposed during the last decade and a half. We review recent results, which cannot be presented in detail due to space limitations. The ensemble configurations considered in this paper are: linear arrays, tori, binary trees, shuffle-exchange, perfect shuffle, and boolean cube networks.

For the description of the algorithms we often refer to the graph model of elimination methods introduced by Parter [29]. The graph of an N by N matrix has N vertices and a directed edge for each nonzero matrix element. The elimination of each vertex requires only a few operations, some of which are independent. The removal of a directed edge can be considered as a primitive operation. Hence, the elimination of a vertex in a tridiagonal system involves only two primitive operations, and the main source of concurrency is the independence of primitive operations for the elimination of independent vertices [15]. As a matrix becomes denser (increased fanout of nodes) the potential concurrency in the elimination of a single vertex increases, and conversely the number of vertices that can be eliminated concurrently decreases.

We first study cyclic reduction on ensemble architectures. The cyclic reduction algorithm concurrently performs primitive operations for the elimination of different vertices. We then consider parallel algorithms that for arbitrary banded systems concurrently eliminate multiple vertices in a partially predetermined order, including algorithms such as parallel block cyclic reduction. This algorithm concurrently performs primitive operations for the elimination of different vertices, and sequentially performs primitive operations for the elimination of a single vertex (precisely as parallel cyclic reduction). Modified systolic algorithms for ensemble architectures are described next. This class of algorithms concurrently performs primitive operations for the elimination of a single vertex. This form of concurrency is very limited for a tridiagonal system, but is the only form of concurrency in elimination methods for dense matrices. Finally a fast banded systems solver is presented. It concurrently performs primitive operations both for the elimination of a single vertex and for multiple vertices.

2. Tridiagonal system solvers

The solution of an irreducible tridiagonal system of N equations and R right hand sides, AX=Y, can be performed in $2\log_2 N$ steps by odd-even cyclic reduction [4] or recursive doubling [24] [40]. For a comparison of the arithmetic complexity of the two methods see Stone [41]. Hockney [14] presents a modified version of cyclic reduction that solves a tridiagonal system in $\log_2 N$ arithmetic steps, at the expense of performing a total of $N\log_2 N$ operations (instead of N). Wang [42] has studied the solution of tridiagonal systems in the context of vector machines. Wang partitions the equations into subsets, applies a variation of Gauss-Jordan elimination to each subset independently, and solves a reduced tridiagonal system having a number of equations that is equal to the number of partitions. The reduced system is also solved by Gaussian elimination. The communication and arithmetic complexity of cyclic reduction and partitioning methods for the solution of tridiagonal systems of N equations on K processor ensembles, $N \geq K$, are analyzed by Gannon and vanRosendale [8] and Johnsson [20]. Ensemble configurations treated in the latter reference are linear arrays, 2-dimensional meshes, binary trees, shuffle-exchange, perfect shuffle and boolean cube networks. The algorithms have distributed control.

In order that the asymptotic execution time on an ensemble architecture of N nodes be of order $O(\log_2 N)$ it is necessary that the mapping of data and computations on to the nodes of the ensemble can be done such that the communication complexity is of order $O(\log_2 N)$. This fact excludes linear arrays and 2-dimensional meshes as candidate configurations [9], since their diameter is N and $2\sqrt{N}$ respectively, and at least one global communication is necessary. The diameter of a binary tree is $2(\log_2 N-1)$, of a shuffle-exchange network $2\log_2 N-1$, and of a boolean cube $\log_2 N$. Nodes in a shuffle-exchange network can be labeled such that every even node is connected to the following odd node (exchange edges), and a node with address obtained by a one step cyclic shift of another node's address is connected to that node (shuffle edges). In a boolean cube, nodes with addresses that differ in precisely one bit are interconnected.

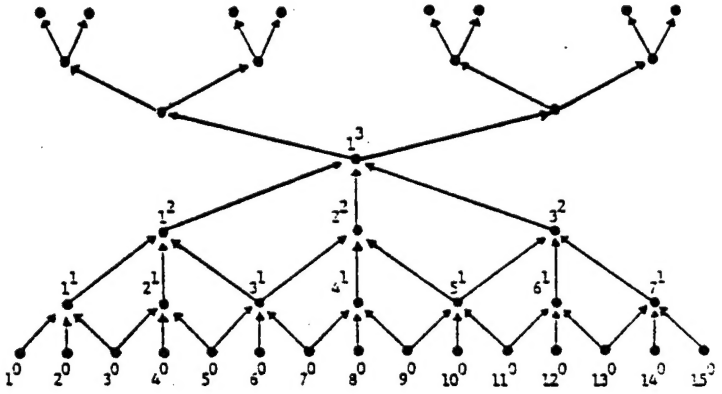

Figure 2-1: A computation graph for odd-even cyclic reduction

For the mapping of odd-even cyclic reduction on to ensemble nodes we start with its computation graph, Figure 2-1. An inorder mapping of equations to nodes in a complete binary tree of matching size yields a complexity of the desired order [22] [8]. A binary tree can be embedded in a shuffle-exchange network such that proximity is preserved. One such embedding is obtained by labeling the nodes of the binary tree in breadth-first order, and identifying nodes so labeled with nodes in the shuffle-exchange network. For the boolean cube an algorithm with distributed control and a communication complexity of order $O(\log_2 N)$ can be devised by embedding the equations in the cube according to a *binary-reflected* Gray code [30]. With such an embedding only 2 routing steps are required for each reduction step. Furthermore, each node can locally determine with which node to exchange what information from knowledge of its own address and the reduction step being processed [20]. Successive reduction steps can be performed in successively lower dimensional subcubes by properly performed local exchanges, as indicated in Figure 2-2. Each reduction step involves only nearest neighbor communication.

One equation per node is not a very realistic assumption for a large system of equations. For a binary tree one may attempt to perform the mapping in two stages; first on to an abstract binary tree of matching size, and then onto the processor tree by folding the abstract tree on to itself a number of times. Such a mapping yields a poor distribution of the computations [20]. The root receives twice as many

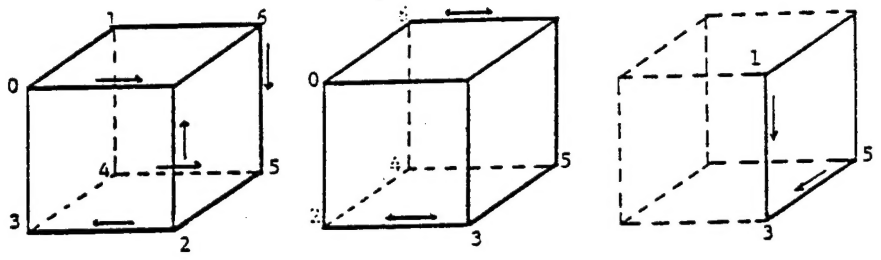

Figure 2-2: Odd-even cyclic reduction on a boolean cube

equations as the other nodes. Moreover, half of the tree nodes become inactive after the first reduction step, a quarter after the second, and so on. Partitioning the sets of equations into P subsets of consecutively indexed equations, or one-dimensional domain decomposition, and mapping subsets in inorder to the P processor tree yields a better result, but the communication complexity is of order $O((\log_2(N/P))\log_2 P+\log_2 P)$, which for $N/P \gg 1$ can be reduced to the order $O(\log_2(N/P)+\log_2^2 P)$ by using a proximity preserving loop embedding [32].

Combining a proximity preserving mapping for the first $\log_2(N/P)$ steps followed by an inorder mapping for the last $\log_2 P$ reduction steps reduces the complexity to order $O(\log_2 N)$, since the conversion between the two mappings can be carried out in $\log_2 P$-3 steps [20]. This is the minimum order possible for cyclic reduction, but not the minimum possible for the solution of a tridiagonal system on a P processor ensemble. The lower bound for the communication complexity is of order $O(\log_2 P)$ for an ensemble having a diameter proportional to $\log_2 P$. Partitioning the set of equations into subsets of consecutively indexed equations, mapping the subsets to processors in inorder, and employing Gaussian elimination in each partition concurrently, followed by cyclic reduction for the reduced system yields an algorithm with a communication complexity of minimum order [20] [8].

For the boolean cube the embedding problem is much simplified, since all required data interactions can be accomplished with at most 2 routing steps with a (fixed) Gray code embedding of subsets [20].

On a linear array cyclic reduction may be of a lower complexity than Gaussian elimination [20]. Both methods are linear in the number of processors in the array, Gaussian elimination because it is inherently sequential and cyclic reduction because of the limited communication capability of the linear array. If the overhead in communication and arithmetic operations is ignored, cyclic reduction is of a lower computational complexity than Gaussian elimination if the communication and arithmetic bandwidths are the same. Conversely, if the arithmetic bandwidth is an order of magnitude higher than the communication bandwidth, then Gaussian elimination is of a lower complexity. Both these results hold independent of the number of right hand sides and the size of the linear array. For intermediate ratios of

the communication and arithmetic bandwidths the method of lowest complexity depends on the number of right hand sides and the size of the array and the number of equations.

For multiple right hand sides it is in most cases preferable to partition the ensemble into subensembles, such that there is one right hand side per ensemble. The communication complexity decreases without a corresponding increase in the arithmetic complexity [20].

Truncation of the reduction process in a maximally concurrent implementation offers a reduction in the computational complexity that is proportional to the extent of the truncation. Hence, truncating the reduction process results in a much more significant reduction in computational complexity than in a sequential implementation, since half of the arithmetic operations are incurred in the first reduction step, a quarter in the second, etc.

3. Concurrent elimination of multiple vertices

Lawrie and Sameh [27] suggest that the system of equations of bandwidth 2m+1, (a_{ij}=0 for $|i-j| > m$) be partitioned into P subsets of consecutively indexed equations, and that in principle Gauss-Jordan elimination be performed concurrently in all partitions. No interpartition communication is required for this phase of the computations. For symmetric matrices a factorization method taking advantage of symmetry may be used in conjunction with linear recurrence solvers. In order that the partitioned system be block tridiagonal P is constrained by the relation $P \leq \lceil N/m \rceil$.

$$
\begin{array}{cccccccc}
A_0 & B_0 & & & & X_0 & & Y_0 \\
C_1 & A_1 & B_1 & & & X_1 & & Y_1 \\
 & C_2 & A_2 & B_2 & & X_2 & & Y_2 \\
 & & \ddots & \ddots & \ddots & \vdots & & \vdots \\
 & & & C_i & A_i \; B_i & X_i & = & Y_i \\
 & & & & \ddots \; \ddots & \vdots & & \vdots \\
 & & & & C_{P-1} \; A_{P-1} & X_{P-1} & & Y_{P-1}
\end{array}
$$

After this first phase, which concurrently eliminates as many vertices as there are partitions (given one processor per partition), a block pentadiagonal system of 2m(P-1) equations can be separated out from the system, and solved in a second phase [27]. The remaining variables are obtained through backsubstitution in a third phase. The second phase requires interpartition communication, whereas the third phase is local to a partition. By allowing communication between adjacent partitions for the backward elimination on the last m equations in each partition it is possible to arrive at a block tridiagonal system of mP equations, instead of the block pentadiagonal system, in the same number of arithmetic operations [21]. The structure of the system after phase 1 is:

$$
\begin{array}{ccccccc}
G_{i_1} & & E_{i_1} & & 0 & & \\
G_{i_2} & I_{q-m} & E_{i_2} & 0_{q-m} & 0 & & \\
G_{i_3} & & E_{i_3} & & 0 & & \\
G_{i_4} & 0 & E_{i_4} & 0 & F_{i_4} & & \\
 & & G_{(i+1)_1} & & E_{(i+1)_1} & & 0 \\
 & 0_{q-m} & G_{(i+1)_2} & I_{q-m} & E_{(i+1)_2} & 0_{q-m} & 0 \\
 & & G_{(i+1)_3} & & E_{(i+1)_3} & & 0 \\
 & 0 & G_{(i+1)_4} & 0 & E_{(i+1)_4} & 0 & F_{(i+1)_4}
\end{array}
$$

The choice of method with respect to computational complexity for the solution of the reduced block tridiagonal system depends on the ensemble configuration and the ratio of the communication and arithmetic bandwidths. The communication and arithmetic complexity for the solution of the reduced system by Gaussian elimination on a linear array is analyzed by Lawrie and Sameh [27]. Dongarra and Sameh [5] investigate the computational complexity of the solution of the reduced system by block Jacobi iterations and preconditioned conjugate gradient methods. In [21] we show that even on a linear array the complexity of block cyclic reduction for the reduced system is lower than that of Gaussian elimination under a variety of conditions. The block size at which cyclic reduction is of a lower complexity than Gaussian elimination decreases with the number of systems to be solved, and a decreasing ratio of the communication and arithmetic bandwidths. For instance, with a ratio of 1, block cyclic reduction is of a lower complexity for all array sizes and matrix bandwidths. With the communication bandwidth 2 orders of magnitude lower than the arithmetic bandwidth, block cyclic reduction becomes of a lower complexity for a matrix bandwidth of about 60 and an array size of about 128 - 256 nodes for 1 right hand side. For 10 right hand sides block cyclic reduction is of a lower complexity than block Gaussian elimination, if the matrix bandwidth is larger than about 50 and the array has more than 64 - 128 nodes.

For ensembles configured as binary trees, shuffle-exchange, perfect shuffle and boolean cube networks, block cyclic reduction can take advantage of the reduced diameter of these networks. The complexity of the partitioning method for a symmetric matrix A is $7m^2N/P+(\beta m+\gamma\alpha)m^2\log_2 P$ where α is the ratio between the arithmetic and the communication bandwidths, and β is 26/3 for a boolean cube and 76/3 for a binary tree. The value of γ is 3 for a boolean cube and 2 for a binary tree. The minimum complexity is of order $O(m^3+m^3\log_2(N/m))$ for the binary tree and boolean cube networks and the corresponding number of processors is $\delta N/m$ with $\delta < 0.81$. For Gaussian elimination on a linear array the minimum complexity is of the order $O(m^2\sqrt{Nm})$ and the corresponding number of processors is of order $O(\sqrt{N/m})$ [27].

The maximum speed-up is of order $O(\sqrt{N/m})$ for a linear array, and of order $O(N/(m+m\log_2(N/m)))$ for a binary tree, shuffle-exchange, perfect shuffle and boolean cube network. For $m \ll N$ the speed-up is

of the maximum possible order for the latter ensemble configurations, but for m≈N essentially only one vertex at a time can be eliminated, and the speed-up is low.

4. Concurrent elimination of a single vertex

Systolic algorithms for the solution of banded systems on ensembles forming 2-dimensional meshes of a size that matches the bandwidth are described by Kung [26] for Gaussian elimination, by Johnsson for Cholesky's, Crout's and Doolittle's methods [17], and for Householder's method [18]. Systolic algorithms for Given's method are described by Ahmed et.al. [1], Heller and Ipsen [12], and Johnsson [19]. The above algorithms concurrently perform all the primitive operations for the elimination of a single vertex. For large bandwidths an array of matching size may not be feasible. Systolic algorithms for Gaussian elimination on banded systems of a bandwidth exceeding the array dimensions are described by Johnsson [16], and for a variety of problems by Heller [11].

Banded system solvers of the systolic type offer a linear speed-up, but are devised for operations of a very fine grain (corresponding to individual matrix elements), and the bulk of the matrix is assumed to be stored outside the array. In an ensemble architecture the entire matrix may be stored in the ensemble, as well as the right hand sides and the result. For the description and the analysis of ensemble architecture algorithms that concurrently eliminate a single vertex it is convenient to introduce the concepts of *operational windows* and *computational windows*. We define an *operational window* to be the set of elementary operations that corresponds to a higher level operation and that can be performed concurrently. For vertex elimination algorithms we define an *operational window* to correspond to the elimination of 1 vertex. A *computational window* is defined by the subset of operations of an operational window that can be performed concurrently by the ensemble [16]. If the array size matches the bandwidth, then the operational and computational windows coincide. For a small array and a wide banded matrix the operational window is partitioned into a number of computational windows, and the execution time increased accordingly. We describe the data and control structures for Gaussian elimination in such a case in [16].

With the entire matrix stored within the ensemble the data movement can be reduced compared to the common systolic algorithms by using a *dual* algorithm, in which the operational window moves over the processor addresses and the data is stationary, instead of the converse. The algorithms that are each other's dual can be obtained through space-time transformations of each other. In the case where the size of the operational window exceeds the ensemble size, the computational windows for each operational window have to be ordered in time. The computations can still be carried out in a fine grain mode or be grouped together to create operations of a larger grain size. Such grouping only affects the complexity if there is an overhead in either communication or arithmetic operations that is not linear in the data. The order of the complexity remains unchanged.

For ensemble configurations in the form of tori and boolean cubes it is often conceptually convenient to regard the matrices as being stored in a 2-dimensional array that is embedded in the ensemble. For ensembles configured as tori the number of nodes in each dimension of the array is taken to be the same as the number of nodes in the ensemble in the corresponding dimension. For ensembles configured as boolean cubes the address space is partitioned into halves, with half of the address space encoding column indices and the other half encoding the row indices of the array. A direct binary encoding can be used. However, we choose to use a binary-reflected Gray code [30] encoding, since such an encoding preserves proximity.

The matrices A, X and Y of the system AX=Y are partitioned such that there are P partitions in each dimension with $\lceil N/m \rceil \leq P \leq N$. The partitioned system consists of blocks of size s by s, where $s=\lceil N/P \rceil$ for some blocks and $s=\lfloor N/P \rfloor$ for other blocks, $r=\lceil (m+1)/s \rceil$, and $t=\lceil R/s \rceil$.

$$
\begin{array}{llllll}
A_{00} & A_{01} & A_{0r-1} & & & \\
A_{10} & A_{11} & \cdot & A_{1r} & & \\
 & \cdot & \cdot & \cdot & & \\
A_{r-1\,0} & A_{r-1\,1} & \cdot & \cdot & A_{r-1\,2r-2} & \\
 & A_{r\,1} & \cdot & \cdot & \cdot & A_{r\,2r-1} \\
 & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & & A_{P-2\,P-r-3} & \cdot & \cdot & A_{P-2\,P-1} \\
 & & & A_{P-1\,P-r-2} & \cdot & A_{P-1P-1}
\end{array}
\begin{array}{llll}
X_{00} & \cdot & X_{0t-1} & \\
X_{10} & \cdot & X_{1t-1} & \\
 & & & \\
 & & & \\
 & & & \\
 & & & \\
 & & & \\
 & & & \\
X_{P-1\,0} & \cdot & X_{P-1t-1} &
\end{array}
=
\begin{array}{llll}
Y_{00} & \cdot & Y_{0t-1} & \\
Y_{10} & \cdot & Y_{1t-1} & \\
 & & & \\
 & & & \\
 & & & \\
 & & & \\
 & & & \\
 & & & \\
Y_{P-1\,0} & \cdot & Y_{P-1\,t-1} &
\end{array}
$$

With the entire matrix stored within the ensemble multiple matrix elements have to be identified with an ensemble node. We consider *consecutive* and *cyclic* storage of the matrix and the right hand sides. In *consecutive* storage all elements of a block matrix are identified, and so are blocks modulo r, i.e., array node $(p,q) \in \{0,1,...,r-1\} \times \{0,1,...,r-1\}$ stores elements (i,j) of the concatenated matrix AY such that $p=\lfloor i/s \rfloor \text{mod} r$ $i=\{0,1,...,N-1\}$ and $q=\lfloor j/s \rfloor \text{mod} r$ $j=\{0,1,...,N+R-1\}$. The number of nodes is r^2, $1 \leq r \leq (m+1)$. In *cyclic* storage the matrix elements (i,j) are identified with node (p,q) if p=imodr, and q=jmodr. The number of processors is s^2, $1 \leq s \leq (m+1)$. The two storage schemes are illustrated in Figure 4-1. The apparent difference in granularity suggested by the two storage forms may vanish if the granularity of operations is optimized.

Using Gaussian elimination the pivot row is communicated to every other row below the diagonal with a nonzero element in the pivot column, and the pivot column (the elements of the factors in the factored form of the inverse) to every other column with a higher column index, assuming that pivoting is done on the diagonal elements. A complexity analysis of the modified systolic algorithm yields $\alpha m(m+R)N/K_c+\beta(m+R)N/\sqrt{K_c}+\gamma(m+R)/\sqrt{K_c}+\delta\sqrt{K_c}$, where the constants are multiples of the time for

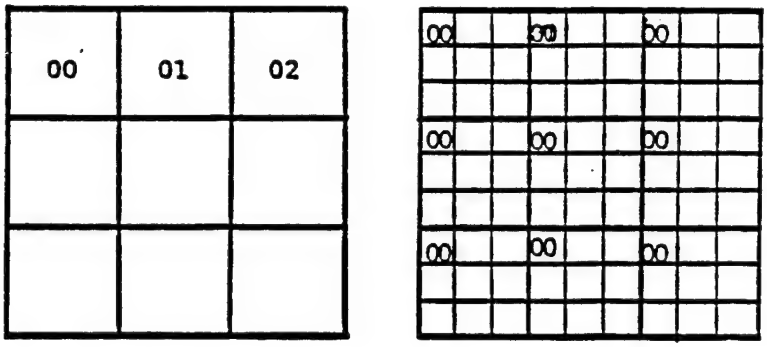

Figure 4-1: Consecutive and cyclic storage of a banded matrix

an arithmetic operation or a communication action. The multiples are small and can be found by a straightforward but tedious analysis [23]. For the special case in which all arithmetic operations require the same time as the communication of a floating-point number the high order terms of the complexity estimate are $3N+(m(m+2(R+1))+R+2)N/K_c - 2m(m(2/3m+R+2)+R+2)/K_c + \phi(m,1/\sqrt{K_c}) + \delta\sqrt{K_c}$.

The speed-up of the highest order term is linear and the optimum value of K_c is of order $O(m(mN^2)^{1/3})$, but $K_c \leq m^2$, i.e., $K_{copt}=\eta m^2$ where η depends on the ratio between the arithmetic and communication bandwidths.

The algorithms described in this section have a linear speed-up for up to m(m+R) processors (possibly 2m(m+R) processors, [6]). For $m \approx N$ the speed-up is good, but it is poor for small values of m. The speed-up characteristics are the inverse of those of the partitioning method.

5. A fast banded system solver

By combining an algorithm for the concurrent elimination of a single vertex with an algorithm for the concurrent elimination of multiple vertices an algorithm is obtained that yields a speed-up of up to order $O(mN/(1+\log_2(N/m)))$ for ensembles configured as clusters of processors with intracluster connections forming tori or boolean cubes, and intercluster connections forming binary trees, shuffle-exchange, perfect shuffle or boolean cube networks. Hence, the speed-up is of order $O(N^2)$ for m of order $O(N)$, and of order $O(N/\log_2 N)$ for $m \ll N$, i.e., of the maximum possible order, and the same as for the tridiagonal system solver described previously.

The two types of algorithms can be combined in the obvious manner, i.e., the system is partitioned for the concurrent elimination of multiple vertices, and each partition is assigned to a distinct cluster of processors. The elimination operations within each partition are carried out by an algorithm exploiting the potential concurrency in the elimination of a single vertex. Note that the algorithms that concurrently eliminate multiple vertices require more operations than the algorithms that perform the vertex elimination in the order defined by the banded matrix, due to fill-in. For instance, cyclic reduction

requires about twice the number of operations of Gaussian elimination performed in perfect elimination order [31].

Phase 1 can be based on Gaussian elimination or a symmetric factorization method if A is symmetric. Phase 2 of the algorithm for the concurrent elimination of multiple vertices requires interpartition communication whether block Gaussian elimination or block cyclic reduction is used. For an irreducible system global communication is needed. For block Gaussian elimination communication in the form of a path is required, whereas block cyclic reduction can be mapped efficiently on to binary trees, shuffle-exchange, perfect shuffle and boolean cube networks [20] [23]. The communication is always between corresponding processors in different clusters. Phase 3 is local to a cluster.

From the previous sections it follows that with P clusters of K_c processors each, with intracluster connections forming a torus or a boolean cube, and intercluster connections forming a binary tree, shuffle-exchange, perfect shuffle, or a boolean cube the communication and arithmetic complexity is of order $O(m(m+R)N/(PK_c)+m^2(m+R)\log_2 P/K_c)$. Clearly, the complexity is minimized if K_c is maximized, i.e., K_c corresponds to the size of an operational window. Furthermore, the optimum value of P is of order $O(N/m)$, regardless of K_c. The optimization with respect to P and K_c can be performed independently. The speed-up is proportional to K_c, but sublinear in P. Hence, a boolean cube should be partitioned for maximum cluster size.

For a linear array there is a complexity term linear in P. This term is entirely due to communication in block cyclic reduction, and is of second order in the matrix bandwidth. The term that is logarithmic in P is of third order in the matrix bandwidth. For Gaussian elimination there is no logarithmic term in P and the term linear in P is of third order in the matrix bandwidth, but has a smaller constant of proportionality than the corresponding arithmetic term in block cyclic reduction. Our estimates of the computational complexity for the solution of the banded system on a linear array using Gaussian elimination for the reduced system is $(N/P-m)m(2m+3R)/K_c+m^2(13m/3+7R/2)(P-1)/(4K_c)$ [23]. The speed-up is again linear in K_c. The optimum number of clusters is of order $O(\sqrt{N/m})$. If block cyclic reduction is used for the solution of the reduced system on the linear array, the complexity instead becomes $(N/P-m)m(2m+3R)/K_c+m(m+R)(P-1)/K_c+ +m^2(14m/3+3R)(\log_2 P-1)/K_c$. The speed-up is linear in K_c. The values of m and P for which block cyclic reduction is of a lower complexity than block Gaussian elimination depend on the ratio of the communication to arithmetic bandwidths.

6. Summary

We have presented parallel algorithms for the concurrent solution of band matrix problems on ensemble architectures. We have ignored numerical aspects (pivoting), which is justified for symmetric positive definite matrices. The algorithms exploit potential concurrency in the elimination of a single vertex, in the elimination of different vertices, or both. Moreover, the algorithms are communication efficient in that the data and computations are mapped on to nodes in regularly configured ensembles such that communication is local to a very large extent.

The speed-up of the algorithms exploiting the potential concurrency in the elimination of a single vertex is linear in the number of ensemble nodes for linear arrays, and 2-dimensional meshes (and boolean cubes). The complexity of the band matrix algorithms exploiting concurrency in the elimination of a single vertex is of order $O(m^2N/K_c)$, where K_c is the number of nodes in the ensemble. The speed-up for algorithms exploiting the potential concurrency in the elimination of multiple vertices is sublinear, indeed at best logarithmic in the number of ensemble nodes because of limited fan-in. For ensembles configured as binary trees, shuffle-exchange, perfect shuffle and boolean cube networks, the minimum complexity attained by the algorithms is proportional to the logarithm of the problem size, including communication time. The complexity of the algorithms that exploit the concurrency both in the elimination of a single vertex and in the elimination of different vertices is of order $O(m^2N/(PK_c)+m^3\log_2P/K_c)$. The minimum value of this complexity is $O(m+m\log_2(N/m))$ for K_c of order $O(m^2)$ and P of order $O(N/m)$. The only communication performed in each step of the algorithms is between adjacent processors.

Acknowledgement

Many thanks go to Andrea Pappas for her assistance with the manuscript.

The support of the Office of Naval Research under contract N00014-84-K-0043 with Yale University is gratefully acknowledged.

References

[1] Ahmed H.M., Delosme J.-M., Morf M.
Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing.
Computer 15:65-82, January, 1982.

[2] Batcher K.E.
Sorting Networks and Their Applications.
In *Spring Joint Computer Conference*, pages 307-314. IEEE, 1968.

[3] Browning S.A.
The Tree Machine: A Highly Concurrent Computing Environment.
Technical Report 1980:TR:3760, Computer Science, California Institute of Technology, January, 1980.

[4] Buzbee,B.L., Golub, G.H., Nielson, C.W.
On Direct Methods for Solving Poissons's Equations.
SIAM J. Numer Anal 7(4):627-656, December, 1970.

[5] Dongarra J.J., Sameh A.H.
On Some Parallel Banded System Solvers.
Technical Report ANL/MCS-TM-27, Argone National Laboratories, Mathematics and Computer Science Division, 1984.

[6] Evans D.J., Hatzopoulos M.
The solution of certain banded systems of linear equations using the folding algorithm.
Computer Journal 19:184-187, 1976.

[7] Flynn M.J.
Very High-Speed Computing Systems.
Proc. of the IEEE 12:1901-1909, 1966.

[8] Gannon D., Van Rosendale J.
On the Impact of Communication Complexity in the Design of Parallel Numerical Algorithms.
Technical Report ICASE Report 84-41, NASA Langley Research Center, August, 1984.

[9] Gentleman W.M.
Some Complexity Results for Matrix Computations on Parallel Processors.
J. ACM 25(1):112-115, January, 1978.

[10] Gottlieb A., Grishman R., Kruskal C.P., McAuliffe K.P., Rudolph L., Snir M.
The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer.
IEEE Trans. Computers C-32(2):175-189, 1983.

[11] Heller D.
Partitioning Big Matrices for Small Systolic Arrays.
VLSI and Modern Signal Processing.
Prentice-Hall, 1985, pages 185-199.

[12] Heller D. E., Ipsen I.C.F.
Systolic Networks for Orthogonal Equivalence Transformations and Their Applications.
In P. Penfield Jr (editor), *Proceedings, Advanced Research in VLSI*, pages 113-122. Artech House, 1982.

[13] Hillis W.D.
The Connection Machine: A Computer Architecture Based on Cellular Automata.
Physica 10D :213-228, 1984.

[14] Hockney R.W., Jesshope C.R.
Parallel Computers.
Adam Hilger, 1981.

[15] Johnsson, S.L.
Gaussian Elimination on Sparse Matrices and Concurrency.
Technical Report 4087:TR:80, Caltech Computer Science Department, December, 1980.

[16] Johnsson, S.L.
Computational Arrays for Band Matrix Equations.
Technical Report 4287:TR:81, Computer Science, California Institute of Technology, May, 1981.

[17] Johnsson S.L.
VLSI Algorithms for Doolittle's, Crout's and Cholesky's Methods.
In *International Conference on Circuits and Computers 1982, ICCC82*, pages 372-377. IEEE, Computer Society, September, 1982.

[18] Johnsson S.L.
A Computational Array for the QR-method.
In P. Penfield, Jr. (editor), *Proc., Conf. on Advanced Research in VLSI*, pages 123-129. Artech House, January, 1982.

[19] Johnsson S. L.
Pipelined Linear Equation Solvers and VLSI.
In *Microelectronics '82*, pages 42-46. Institution of Electrical Engineers, Australia, May, 1982.

[20] Johnsson S.L.
Odd-Even Cyclic Reduction on Ensemble Architectures and the Solution Tridiagonal Systems of Equations.
Technical Report YALE/CSD/RR-339, Department of Computer Science, Yale University, October, 1984.

[21] Johnsson S.L.
Solving Narrow Banded Systems on Ensemble Architectures.
Technical Report YALEU/CSD/RR-343, Dept. of Computer Science, Yale University, November, 1984.

[22] Johnsson,S.L.
Cyclic Reduction on a Binary Tree.
Computer Physics Communications ():, 1985.
Presented at the 1984 Conf. on Vector and Parallel Processors in Computational Science II.

[23] Johnsson S.L.
Fast Banded Systems Solvers for Ensemble Architectures.
Technical Report YALEU/CSD/RR-379, Department of Computer Science, Yale University, March, 1985.

[24] Kogge P.M., Stone H.S.
A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations.
IEEE Trans. Computers C-22(8):786-792, 1973.

[25] Kuck D.J., Lawrie D.H., Cytron R., Sameh A., Gajski D.D.
The Architecture and the Programming of the Cedar System.
Technical Report, Laboratory for Advanced Supercompuetrs, Dept. of Computer Science, University of Illinois, August, 1983.

[26] Kung H.T., Leiserson C.L.
Algorithms for VLSI Processor Arrays.
Introduction to VLSI Systems.
Addison-Wesley, 1980, pages 271-292.

[27] Lawrie D.H.,Sameh A.H.
The Computational Complexity and Communication Complexity of a Parallel Banded System Solver.
ACM Trans. Math. Software 10(2):185-195, June, 1984.

[28] Mead C.A., Conway L.
Introduction to VLSI Systems.
Addison-Wesley, 1980.

[29] Parter S.
The use of linear graphs in Gaussian elimination.
SIAM Review 3(2):119-130, 1961.

[30] Reingold E.M., Nievergelt J., Deo N.
Combinatorial Algorithms.
Prentice Hall, 1977.

[31] Rose D.J.
A Graph-Theoretic Study of the Numerical Solution of Sparse Positive Definite Sytsems of Linear Equations.
In *Graph Theory and Computing*, pages 183-217. Academic Press, 1971.

[32] Rosenberg A.L., Snyder L.
Bounds on the Costs of Data Encodings.
Mathematical Systems Theory 12:9-39, 1978.

[33] Schwartz J.T.
Ultracomputers.
ACM Trans. on Programming Languages and Systems 2:484-521, 1980.

[34] Seitz C.L.
Self-Timed VLSI Systems.
In *Proc. of the Caltech Conference on Very Large Scale Integration*, pages 345-355. Computer Science, California Institute of Technology, 1979.

[35] Seitz C.L.
The Cosmic Cube.
Communications of the ACM 28(1):22-33, 1985.

[36] Sejnowski M.C., Upchurch E.T., Kapur R.N., Charlu D.P.S., Lipovski G.J.
An Overview of the Texas Reconfigurable Array Computer.
In *Proceedings, National Computer Conference*, pages 631-641. IEEE, 1980.

[37] Shaw D.
The NON-VON Supercomputer.
Technical Report, Dept. of Computer Science, Columbia University, August, 1982.

[38] Shaw D.
SIMD and MSIMD Variants of the NON-VON Supercomputer.
Technical Report, Dept. of Computer Science, Columbia University, 1984.

[39] Snyder L.
Introduction to the Configurable Highly Parallel Computer.
Computer 15(1):47-56, 1982.

[40] Stone, H.S.
An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations.
JACM 20(1):27-38, January, 1973.

[41] Stone, H.S.
Parallel Tridiagonal Equation Solvers.
ACM Transactions on Mathematical Software 1(4):289-307, December, 1975.

[42] Wang H.H.
A Parallel Method for Tridiagonal Equations.
ACM Trans. Math. Software 7(2):170-183, June, 1981.

Reproduced by NTIS

National Technical Information Service
Springfield, VA 22161

This report was printed specifically for your order from nearly 3 million titles available in our collection.

For economy and efficiency, NTIS does not maintain stock of its vast collection of technical reports. Rather, most documents are printed for each order. Documents that are not in electronic format are reproduced from master archival copies and are the best possible reproductions available. If you have any questions concerning this document or any order you have placed with NTIS, please call our Customer Service Department at (703) 487-4660.

About NTIS

NTIS collects scientific, technical, engineering, and business related information — then organizes, maintains, and disseminates that information in a variety of formats — from microfiche to online services. The NTIS collection of nearly 3 million titles includes reports describing research conducted or sponsored by federal agencies and their contractors; statistical and business information; U.S. military publications; audiovisual products; computer software and electronic databases developed by federal agencies; training tools; and technical reports prepared by research organizations worldwide. Approximately 100,000 *new* titles are added and indexed into the NTIS collection annually.

For more information about NTIS products and services, call NTIS at (703) 487-4650 and request the free *NTIS Catalog of Products and Services*, PR-827LPG, or visit the NTIS Web site **http://www.ntis.gov**.

NTIS
Your indispensable resource for government-sponsored information—U.S. and worldwide